AEDC-TR-86-45
C-4

Conversion of ARO-1 to an Implicit Navier-Stokes Solver

Karl R. Kneile
Sverdrup Technology, Inc.

December 1986

Final Report for Period October 1, 1983 – October 1, 1986

ARNOLD ENGINEERING DEVELOPMENT CENTER
ARNOLD AIR FORCE STATION, TENNESSEE
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE

APPROVAL STATEMENT

This report has been reviewed and approved.

Matthew C. Towne

MATTHEW C. TOWNE, Capt, USAF
Facility Technology Division
Directorate of Technology
Deputy for Operations

Approved for publication:

FOR THE COMMANDER

[illegible]

LOWELL C. KEEL, Colonel, USAF
Director of Technology
Deputy for Operations

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT SEE REVERSE OF THIS PAGE
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AEDC-TR-86-45	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Arnold Engineering Development Center — 6b. OFFICE SYMBOL (If applicable) DOT	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Systems Command Arnold Air Force Station, TN 37389-5000	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Arnold Engineering Development Center — 8b. OFFICE SYMBOL (If applicable) DO	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code) Air Force Systems Command Arnold Air Force Station, TN 37389-5000	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. 65807F; PROJECT NO; TASK NO; WORK UNIT NO
11. TITLE (Include Security Classification) SEE REVERSE OF THIS PAGE	

12. PERSONAL AUTHOR(S)
Kneile, K. R., Sverdrup Technology, Inc., AEDC Group

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 10/1/83 TO 10/1/86	December 1986	32

16. SUPPLEMENTARY NOTATION
Available in Defense Technical Information Center (DTIC)

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB GR	
20	04		Navier-Stokes solver
09	02		fluid dynamic equations

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The purpose of this report is to describe the develoΟment of an implicit Navier-Stokes solver (KM1) based upon modification of an existing explicit Euler code known as ARO-1. The report narrates the steps of the conversion, relates the lessons learned alonq the way, and highlights the algorithm's advantages and shortcomings. The goal upon inception of this work was to develop an efficient Navier-Stokes solver, capable of solving complex three-dimensional (3-D) internal flow problems in the Engine Test Facility of AEDC. It was to be modeled after the ARO-1 code because of ARO-1's established flexibility and robustness. While the KM1 code successfully solved the flow problems that were tried, it does not have the anticipated robustness that was expected. Considerable user interaction was required to obtain solutions for certain complex 3-D problems. The code has taken a backseat to a more robust code recently acquired. The KM1 code is still in its developmental form, and there are no plans for converting the code to a production format.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☒ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL W. O. Cole	22b. TELEPHONE NUMBER (Include Area Code) (615) 454-7813 — 22c. OFFICE SYMBOL DOS

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

3. DISTRIBUTION/AVAILABILITY OF REPORT

Approved for public release; distribution is unlimited.

11. TITLE

Conversion of ARO-1 to an Implicit Navier-Stokes Solver

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

PREFACE

The work reported herein was conducted by the Arnold Engineering Development Center (AEDC), Air Force Systems Command (AFSC). The Air Force Program Manager was Dr. Keith Kushman. The results were obtained by Sverdrup Technology Inc., AEDC Group, operating contractor for the propulsion testing facilities at the AEDC, AFSC, Arnold Air Force Station, Tennessee 37389. The research was conducted in the Engine Test Facility (ETF) during the period October 1, 1983 through October 1, 1986 under AEDC Project Numbers DA88EW and DB84EW, and the manuscript was submitted for publication November 4, 1986.

The author acknowledges the contribution of Professor R. W. MacCormack for his contributions to the work described in this report.

CONTENTS

ILLUSTRATIONS

1.0 INTRODUCTION

The purpose of this report is to describe the development of an implicit Navier-Stokes solver (KM1) based upon modification of an existing explicit Euler code known as ARO-1. The report narrates the steps of the conversion, relates the lessons learned along the way, and highlights the algorithm's advantages and shortcomings. The goal upon inception of this work was to develop an efficient Navier-Stokes solver, capable of solving complex three-dimensional (3-D) internal flow problems in the Engine Test Facility at AEDC. It was to be modeled after the ARO-1 code because of ARO-1's established flexibility and robustness. While the KM1 code successfully solved the flow problems that were tried, it does not have the anticipated robustness that was expected. Considerable user interaction was required to obtain solutions for certain complex 3-D problems. The code has taken a backseat to a more robust code recently acquired. The KM1 code is still in its developmental form, and there are no plans for converting the code to a production format.

2.0 GOVERNING EQUATIONS

The governing fluid dynamic equations for compressible, viscous flow consist of the mass, momentum, energy, and state equations collectively termed the Navier-Stokes equations. These equations can be written as

$$\frac{\partial U}{\partial t} + \nabla \cdot F = 0 \qquad (1)$$

where the flux vector F is

$$F = \begin{bmatrix} \varrho u \\ \varrho uu + PI - \tau \\ (E + P)u + u \cdot \tau - k\nabla T \end{bmatrix} \qquad (2)$$

and the dyadic τ represents the viscous portion of the stress tensor given by

$$\tau = 2\mu \widehat{\nabla u} + \lambda (\nabla \cdot u) I \qquad (3)$$

The quantity U consists of the conservative variables, (ϱ, ϱu, and E). The dyadic $\widehat{\nabla u}$ is the symmetric part of ∇u, and I represents the unit dyadic. The flow is considered laminar with the viscosity calculated using Sutherland's formula

$$\frac{\mu}{\mu_r} = \frac{T_r + 198.6}{T + 198.6} \left(\frac{T}{T_r}\right)^{3/2} \qquad (4)$$

and λ is obtained from Stokes hypothesis

$$\lambda = -\frac{2}{3}\mu \tag{5}$$

Using the equation of state (calorically and thermally perfect), pressure can be determined from

$$P = (\gamma - 1)\left[E - \frac{1}{2}\varrho u \cdot u\right] \tag{6}$$

The equations are nondimensionalized as follows:

$$\begin{array}{lll} \bar{\varrho} = \varrho/\varrho_\infty & \bar{P} = P/\varrho_\infty a_\infty^2 & \bar{x} = x/L \\ \bar{u} = u/a_\infty & \bar{E} = E/\varrho_\infty a_\infty^2 & \bar{t} = a_\infty t/L \\ \bar{T} = T/T_\infty & \bar{\mu} = \mu/\mu_\infty & \bar{\lambda} = \lambda/\mu_\infty \end{array} \tag{7}$$

As a result of the above definitions, the following relationships can be determined:

$$\begin{aligned} \bar{T} &= \bar{a}^2 = \gamma\bar{P}/\bar{\varrho} \\ \bar{P} &= P/\gamma P_\infty = (\gamma - 1)\left[\bar{E} - \frac{1}{2}\bar{\varrho}\bar{u} \cdot \bar{u}\right] \end{aligned} \tag{8}$$

The resulting nondimensional Navier-Stokes equations are

$$\frac{\partial \bar{U}}{\partial \bar{t}} + \nabla \cdot \bar{F} = 0 \tag{9}$$

with

$$\bar{F} = \begin{bmatrix} \bar{\varrho}\bar{u} \\ \bar{\varrho}\bar{u}\bar{u} + \bar{P}I - \bar{\tau} \\ (\bar{E} + \bar{P})\bar{u} - \bar{u} \cdot \bar{\tau} - \bar{k}\nabla\bar{T} \end{bmatrix} \tag{10}$$

and

$$\begin{aligned} \bar{\tau} &= Re^{-1}\left(2\bar{\mu}\,\nabla\bar{u} + \bar{\lambda}\,\nabla \cdot \bar{u}I\right) \\ \bar{k} &= \bar{\mu}/Re\,Pr\,(\gamma - 1) \end{aligned} \tag{11}$$

The Reynolds number, Re, and Prandtl number, Pr, are given by

$$
\begin{aligned}
\mathrm{Re} &= \varrho_\infty \, a_\infty \, L/u_\infty \\
\mathrm{Pr} &= \mu \, c_p/k
\end{aligned}
\tag{12}
$$

Only nondimensional variables will be used hereafter, and the "bar" notation will be dropped.

It is convenient at times to split the flux vector into inviscid and viscous parts, that is,

$$
F = F^* - G \tag{13}
$$

where

$$
F^* = \begin{bmatrix} \varrho u \\ \varrho uu + PI \\ (E + P)u \end{bmatrix} \tag{14}
$$

and

$$
G = \frac{\mu}{\mathrm{Re}} \begin{bmatrix} 0 \\ \tau \\ u \cdot \tau + \nabla T/(\gamma - 1)\,\mathrm{Pr} \end{bmatrix} \tag{15}
$$

For inviscid flow, $\mu = 0$, $F = F^*$, and the resulting equations are known as the Euler equations. In this case, when no confusion arises and the intent is obvious, the "*" will be dropped, and F will be used as a conventional simplification.

3.0 EXPLICIT EULER CODE

Since the starting point for this effort was an existing Euler code (ARO-1), a brief description of the algorithm will be given here. For additional information concerning ARO-1 the reader is referred to Refs. 1 and 2.

The grid imposed on the computational flow region is topologically equivalent to a regular grid on a cube. The resulting volume elements are hexahedrons with quadrilateral faces. Degenerate (nonhexahedral) volumes may be used; for example, two edges may coincide to form a distorted triangular prism typically found along the axis in axisymmetric grids.

A finite volume algorithm is used with the flow variables being calculated at volume centers. The algorithm is obtained by integrating Eq. (9) over a grid volume and applying the divergence and mean value theorems, giving

$$\frac{\partial U}{\partial t} = -\frac{1}{V}\oint n \cdot F ds \tag{16}$$

V is the volume of the grid cell, n is an outward pointing unit normal, and $\oint$ represents a surface integral over the entire cell. This equation is descretized to obtain

$$\Delta U = -\frac{\Delta t}{V}\sum An \cdot F \tag{17}$$

where the summation is over the six faces of the volume. A is the projected area of the surface in the direction of n (that is, An is an area vector). An explicit MacCormack algorithm is used to evaluate Eq. (17). The MacCormack algorithm is a two-step Runge-Kutta procedure using forward spacial differencing for the first (predictor) step and backward differencing for the second (correct) step, or vice versa. The finite volume analog of this is equivalent to evaluating F in Eq. (17) for a given face, using the value at either the forward or backward volume centers. Forward or backward here refers to the direction of node indexing. For the code in Ref. 1, the Euler equations are solved and $F = F^*$. Since F^* does not contain any derivatives, it can be evaluated using only the values at the specified (forward or backward) centers.

At the boundaries, phantom cells are used for values outside the boundary face. Boundary conditions and neighboring interior values are used to define these phantom (boundary) values.

4.0 VISCOUS TERMS

Conversion of the ARO-1 code to a Navier-Stokes solver requires the evaluation of the viscous terms [Eq. (15)]. In the finite volume formulation, the divergence operator is replaced by the area flux calculations. Since the only derivatives in the Euler equation are those in the divergence operator, the derivatives are conveniently hidden as flux calculations. Viscous terms, however, take the form of second-order derivatives. As in the Euler case, the divergence derivatives are hidden in the flux calculations. But the second-order derivatives reduce to first-order derivative flux terms. Therefore, derivatives of four quantities (the three velocity components and temperature) are required. The derivatives are calculated using a localized coordinate system and then transformed to the global Cartesian system. The method will be described for an ith face (that is, the face between the i, j, kth and the i + 1, j, kth cell centers). Local differencing is easily accomplished as follows. Differencing in the ith direction is done between the i + 1st and ith values. Differencing in the other directions uses j ± 1 and k ± 1 subscripts. For example,

$$
\begin{aligned}
\Delta_i T &= T_{i+1,j,k} - T_{i,j,k} . \\
\Delta_j T &= T_{i^*,j+1,k} - T_{i^*,j-1,k} \\
\Delta_k T &= T_{i^*,j,k+1} - T_{i^*,j,k-1}
\end{aligned}
\tag{18}
$$

The value of i* is either i or i + 1 and corresponds to a forward or backward evaluation in the MacCormack algorithm. It is chosen consistently with the value used for evaluating the Euler flux, F*. Differences are calculated for the four dependent quantities, T, u, v, w, and the Cartesian coordinates, x, y, and z. The Jacobian matrix

$$
J = \begin{bmatrix} \Delta_i x & \Delta_i y & \Delta_i z \\ \Delta_j x & \Delta_j y & \Delta_j z \\ \Delta_k x & \Delta_k y & \Delta_k z \end{bmatrix}
\tag{19}
$$

can be used to transform the local differences to Cartesian derivative estimates. For example, the T derivatives are obtained by

$$
\begin{bmatrix} \Delta T/\Delta x \\ \Delta T/\Delta y \\ \Delta T/\Delta z \end{bmatrix} = J^{-1} \begin{bmatrix} \Delta_i T \\ \Delta_j T \\ \Delta_k T \end{bmatrix}
\tag{20}
$$

All nonderivative calculations are made using a forward/backward evaluation consistent with the inviscid terms.

5.0 IMPLICIT OPERATOR

The conversion from the 1969 explicit algorithm of Refs. 1 and 2 to the 1981 implicit algorithm developed by MacCormack (Ref. 3) is a straightforward extension. The predictor/corrector estimates obtained from the explicit algorithm are used as right-hand sides of the implicit equations. The implicit operator is based upon the inviscid or convection part of the equations. The viscous effects are only included through a diagonal (eigenvalue) modification. See Ref. 3 for the justification/rationale of this algorithm. The primary advantage of the algorithm is its computational efficiency. This is particularly evident in three areas.

1. The implicit systems are block diagonal; other algorithms give block tridiagonal systems. In fact, the unfactored form is triangular. Solution of these systems only requires a backward substitution.

2. The systems are easily converted to scalar form. The diagonalization is actually part of the implicit operator and not added on artificially.

3. The algorithm automatically reverts to an explicit form when CFL criteria allow. It is easy to completely bypass implicit calculations over portions of the flow field when they are not needed. Because of the nature of the grids for most viscous problems, this feature is easily adapted to vector processing.

The following is a description of the 1981 implicit MacCormack algorithm as applied to the 3-D finite volume (ARO-1) code. The implicit operator can be written as

$$[I + \overline{D^1 \cdot F_U} + \overline{D^2 \cdot F_U} + \overline{D^3 \cdot F_U}]\delta U = \Delta U \tag{21}$$

The ΔU are the temporal changes calculated from the explicit algorithm. F_U represents the Jacobian matrices containing the partial of F^* with respect to U. As only the inviscid portion of F is used for the implicit operator, the * notation will be dropped to simplify the equations. The bar notation corresponds to MacCormack's absolute valve, |...|, notation and will be described later. The D^i represents finite volume difference operators in the three directions implied by the grid indices. These operators are calculated in a forward/backward form, corresponding to the directions used in the explicit calculation of ΔU. Since only first-order differencing is used, the equation can be written in a block triangular form and solved using a simple backward substitution algorithm. The factored form of the implicit operator is

$$\begin{aligned}[I + \overline{D^1 \cdot F_U}]\ \delta\tilde{U} &= \Delta U \\ [I + \overline{D^2 \cdot F_U}]\ \delta\tilde{\tilde{U}} &= \delta\tilde{U} \\ [I + \overline{D^3 \cdot F_U}]\ \delta U &= \delta\tilde{\tilde{U}}\end{aligned} \tag{22}$$

The above factored form is an approximation to the unfactored form [Eq. (21)] requiring the dropping of higher-order terms. The order of the D^i in this form may be permuted. The permuting of the D^i gives different answers but they are equal to the order of accuracy of the algorithm. Actual test cases have indicated that cycling through the various permutations gives better answers than a given fixed permutation.

The approximate factorization converts the system from a large 3-D type problem to three separate one-dimensional (1-D) problems to be solved in succession. That is, the solutions of earlier systems are used as right-hand sides of the successive system. The three systems are identical in form and will be described further using the generic form

$$\left[I + \overline{D \cdot F_U}\right]\delta U = \Delta U \tag{23}$$

The difference operator, D·, contains two area vectors from opposite faces in the hexahedral volume (one pair for each $D^{\cdot i}$). These vectors may be denoted as An_A and Bn_B. The n_A and n_B are unit normal vectors, and A and B are the area magnitudes. For notational purposes, n_A and A correspond to the area for which F_U is evaluated using the volume center values. The corresponding forward/backward volume values are used for evaluating F_U associated with n_B and B. Equation (23) can now be written as

$$\left[I + \frac{\Delta t\, A}{V}\overline{n_A \cdot F_U^A}\right]\delta U_A = \Delta U + \frac{\Delta t\, B}{V}\overline{n_B \cdot F_U^B}\,\delta U_B \tag{24}$$

where V is the cell volume and Δt is the time increment. The block bidiagonal structure, requiring only backward substitution, is readily apparent in this form. The δU_B represents known temporal increments from a neighboring volume. Therefore, once the values of ΔU are known at the proper boundary, all the other values can be calculated one by one along a path in the appropriate direction.

The calculation of the barred expression and the solution of Eq. (24) are interrelated through characteristic theory. For notational simplification, n and F_U will be used to denote n_A and F_U^A. The matrix $n \cdot F_U$ can be diagonalized as

$$n \cdot F_U = Q^{-1}\Lambda Q = P^{-1}S^{-1}R^{-1}\Lambda RSP \tag{25}$$

where

$$Q = RSP \tag{26}$$

$$R = \begin{bmatrix} 0 & -n & 1 \\ n & I - nn & n \\ 0 & n & 1 \end{bmatrix} \tag{27}$$

$$S = \begin{bmatrix} a^2 & \phi & 0 \\ \phi & aI & \phi \\ 0 & \phi & \beta \end{bmatrix} \tag{28}$$

$$P = \begin{bmatrix} 1 & \phi & 0 \\ -u & I & \phi \\ \alpha & -u & 1 \end{bmatrix} \tag{29}$$

$$\Lambda = \begin{bmatrix} u_n - a & \phi & 0 \\ \phi & u_n I & \phi \\ 0 & \phi & u_n + a \end{bmatrix} \tag{30}$$

$u_n = u \bullet n, \alpha = u \bullet u/2, \beta = \gamma - 1$, and $a^2 = \gamma P/\varrho$. The inverses are easily shown to be

$$R^{-1} = \begin{bmatrix} 1 & n & 1 \\ -n & I - nn & n \\ 1 & \phi & 1 \end{bmatrix} \begin{bmatrix} 1/2 & \phi & 0 \\ \phi & I & \phi \\ 0 & \phi & 1/2 \end{bmatrix} \tag{31}$$

$$P^{-1} = \begin{bmatrix} 1 & \phi & 0 \\ u & I & \phi \\ \alpha & u & 1 \end{bmatrix} \tag{32}$$

The factored form of the diagonalization allows easy and efficient calculations of the required characteristic transformations. Another advantage of this particular decomposition is that it does not require the arbitrary definition of two unit vectors in the normal plane. We can now define

$$\overline{n \bullet F_U} = Q^{-1} D^{+} Q \tag{33}$$

where

$$D^{+} = (D + |D|)/2 \tag{34}$$

and

$$D = |\Lambda| + \left(\frac{2\gamma\mu A}{Re\ Pr\ \varrho V} - \frac{V}{2A\Delta t} \right) I \tag{35}$$

The absolute values of the matrices D and Λ, namely |D| and |Λ|, are defined to be the matrices of absolute values of their elements. Although the decomposition is based on characteristics from the inviscid equations, the definition of D includes a term containing a viscous parameter to maintain stability. When the local CFL criterion is satisfied, the elements of D are negative and D^+ is null. The system then is identical to the explicit algorithm, that is,

$$\delta U_A = \Delta U \tag{36}$$

Computationally, the system is solved using the following steps:

1. The transferred flux from the neighboring cell is added to ΔU to form the right-hand side

$$\widehat{\Delta U} = \Delta U + \frac{\Delta t\, B}{V} \overline{n_B \cdot F_U^B}\, \delta U_B \tag{37}$$

2. The right-hand side is multiplied by the inverse, calculated using the characteristic decomposition.

$$\delta U = P^{-1} S^{-1} R^{-1} \left[I + \frac{\Delta t\, A}{V} D^+ \right]^{-1} RSP\, \widehat{\Delta U} \tag{38}$$

The center matrix is diagonal and easily inverted. The matrix multiplications are done from right to left. Therefore, all products are of a square matrix times a column matrix form. This is more efficient than multiplying two square matrices.

3. The flux transferred to the next cell is calculated from

$$\frac{\Delta t\, A}{V} \overline{n_A \cdot F_U^A}\, \delta U_A = \widehat{\Delta U} - \delta U_A \tag{39}$$

The above procedure implicitly transfers information from one boundary to another. By sequentially using the three 1-D operators [Eq. (22)], information is passed from three of the six global boundaries to the other three. Like the 1969 explicit algorithm, the 1981 implicit algorithm consists of two steps, predictor and corrector. The final results represent an average of the two steps. Since the differencing (forward/backward) is reversed between the two steps, the implicit operator passes information one way on the predictor and the other on the corrector, resulting in an algorithm that is implicit in both directions.

6.0 BOUNDARY CONDITIONS

The physical boundary conditions are applied in an explicit sense. However, the implicit operator, being a separate part, requires its own boundary conditions. The explicit physical boundary conditions are what actually determine the solution. The boundary conditions for the implicit operator affect the convergence and stability of the algorithm. The one-way transfer of information, which makes the implicit operator so easy to solve, creates problems with choosing suitable boundary conditions. It is this difficulty of handling the implicit operator boundary conditions that causes the most problems with this algorithm.

6.1 BOUNDARY CONDITIONS FOR IMPLICIT OPERATOR

Fortunately, for many boundaries, it is possible to avoid the need for an implicit boundary condition. If the grid spacing normal to the boundary is sufficiently large, the local CFL in the normal direction will turn the implicit operator off in that direction. For inflow and outflow conditions, the spacing in the flow direction is considerably larger than the spacing normal to a wall. It was therefore assumed that at the inflow/outflow boundaries, the CFL normal to the boundaries will allow explicit conditions. Although this might limit the allowable time step, it did not prove to be a serious drawback as the time step was usually limited for other reasons. This restriction does not preclude the compressing of the grid spacing interior to the boundaries as the implicit operator can properly turn itself on whenever the algorithm switches from explicit to implicit.

In order to resolve boundary-layer calculations, the spacing near and normal to walls must be very small, typically orders-of-magnitude smaller than in other parts of the flow field. Because of this, one would expect to be able to run at time steps requiring implicit calculation at the wall boundaries.

When originally proposed in 1981, the algorithm required that the implicit operator pass information towards the wall on the predictor step and away from the wall on the corrector step. Therefore, only the corrector step required implicit boundary conditions. The boundary conditions for the corrector step were obtained by reflecting the outgoing flux from the predictor step back into the flow field. The opposing boundary was not a wall, and the grid spacing was large enough to turn the implicit operator off and use explicit calculations.

When two walls are opposing each other (for example, a nozzle), one of the walls will require implicit boundary conditions for the predictor step. First attempts used reflected outgoing flux from the corrector step as boundary conditions for the next predictor step. Although this approach worked, a more robust method was developed in which the incoming fluxes were assumed to be zero. The outgoing fluxes were immediately reflected and redistributed back into the flow field using the implicit operator for passing information in the other direction. That is, at and near the walls both operators (forward/backward) were used successively. This approach required that the grid spacing somewhere between the two walls be large enough for the implicit operator to turn off. Otherwise this information would be shuttled back and forth between the two walls without ending.

6.2 PHYSICAL BOUNDARY CONDITIONS

These are the boundary conditions that actually determine the solution. They are imposed after the implicit calculations and in certain cases replace values obtained from the implicit operator. Four types of physical boundary conditions have been considered.

The simplest type is a symmetry condition. For this case the values are reflected through the boundary. Phantom points are used for these boundary conditions. The equations are

$$\begin{aligned} \varrho_b &= \varrho \\ E_b &= E \\ (\varrho u)_b &= \varrho u - 2\varrho u \bullet nn \end{aligned} \tag{40}$$

where the b subscript represents the phantom boundary point and the unsubscripted values are evaluated at its interior neighbor.

Next considered are wall conditions. Again, phantom cells are used. Slip walls can be handled using the symmetry condition [Eq. (40)]. Nonslip walls require that the velocity vector vanish at the wall. A zero normal pressure gradient is also assumed. If one further assumes an adiabatic wall, then the conditions are equivalent to

$$\begin{aligned} \varrho_b &= \varrho \\ E_b &= E \\ (\varrho u)_b &= -\varrho u \end{aligned} \tag{41}$$

For a wall with a known temperature, T_w, one uses

$$\begin{aligned} T_b &= 2T_w - T \\ P_b &= P \\ (\varrho u)_b &= -\varrho u \end{aligned} \tag{42}$$

obtaining ϱ_b and E_b from T_b, P_b, and $(\varrho u)_b$.

The most complex boundary conditions are those for inflow and outflow. The original inviscid ARO-1 code used reference plane characteristics for the inflow/outflow conditions.

When these inviscid routines were used for the viscous case, poor results were obtained in the boundary layer, especially at the inflow boundary. This is because the reference plane characteristics algorithm did not include the viscous terms. As a result, a new algorithm was derived, based on characteristic theory, that would allow one to include viscous effects. The new algorithm does not require the interpolation usually required for characteristic algorithms and is very efficient and very robust. The key point of the new algorithm is that it uses results from the interior algorithm and modifies them. Phantom cells are not used. The interior algorithm is modified so that at inflow and outflow boundary points, only inward differencing/evaluation is used. These calculations give the temporal changes at boundary points due to the flow equations, using only points in the flow field (that is, no phantom points). For boundaries with no external influence (for example, supersonic outflow), these calculations are all that are needed. Indeed, the new algorithm will leave these calculations unmodified for these cases. When characteristic theory requires outside influence, these calculations are modified to properly include this influence.

The algorithm is based on the following observations. If Eq. (1) is multiplied by the transformation matrix, Eq. (26), a new system of equations is obtained that is equivalent to the original system. However, the new system has a one-to-one relationship with the compatibility equations from the reference plane characteristic algorithm. In fact, they are the same equations in different form (partial derivatives versus total derivatives). The direction of influence of these equations can be determined from the eigenvalue matrix, Eq. (30).

The new algorithm proceeds as follows. A trivial system of equations,

$$Q\Delta U = Q(\Delta U)^* \tag{43}$$

gives the characteristic information. Here, $(\Delta U)^*$ are the unmodified calculated values from the equations. The sign of the eigenvalues (elements of Λ) is used to determine which equation represents internal information. These equations are retained, and the others are replaced with external boundary conditions.

As pointed out earlier, for supersonic outflow all the equations are kept, and the solution of Eq. (43) is

$$\Delta U = (\Delta U)^* \tag{44}$$

For subsonic outflow, the $u_n - a$ eigenvalue is negative, indicating that it no longer represents internal information. Therefore, the corresponding equation is replaced with an appropriate boundary condition, that is,

$$P = P^* \quad \text{or} \quad \Delta P = P_{new} - P_{old} \tag{45}$$

This new system is then solved for the modified temporal changes.

For subsonic inflow, only one relationship from Eq. (43) is retained. The other four are replaced with boundary conditions. Typically, these are specified total temperature T_0, specified total pressure P_0, and two flow-angle conditions. These conditions work well for inviscid flow regions, but in the boundary layer, P_0 is usually not known a priori. Therefore, the inflow routine includes a boundary-layer region where P is specified to be equal to the value at the edge of the inviscid region, simulating $dP/dn = 0$ through the boundary layer. Two types of flow-angle conditions are used, fixed flow angles and no change in flow angles for the inward grid direction.

7.0 TEST CASES

The code was verified using three test cases from Thomas (Refs. 4 and 5): boundary layer on a flat plate, two-dimensional (2-D) transonic nozzle flow, and transonic nozzle flow for a rectangular converging-diverging 3-D nozzle. The wall boundary-layer calculations produced good agreement with the Blasius boundary-layer solution and with the calculations of Thomas. The 2-D nozzle calculations produced good agreement with measured wall pressures along the symmetry plane of a rectangular nozzle. The results were also in close agreement with the results of Thomas.

The 3-D capabilities of the code were demonstrated with the rectangular converging-diverging nozzle case. The Reynolds number, based on stagnation conditions and throat half-height, was 930,000. The flow was assumed laminar, and the Prandtl number was 0.72. Figure 1 illustrates the nozzle computational grid. Two planes of symmetry (XY and XZ) were used. Figure 2 shows static pressure results along the nozzle wall at the vertical symmetry plane. Figure 3 shows static pressure along the nozzle side wall at the horizontal symmetry plane. The open symbols represent experimental data (Ref. 6), and the solid symbols are the results of the method presented in this report. The solid line shows the results of Thomas.

Although linearized 1-D theory shows the present algorithm to be unconditionally stable, in practice an upper stability limit is prevalent. For the simpler flat-plate and 2-D nozzle cases, a CFL of 200 to 1000 could be routinely used. For the 3-D nozzle, the CFL limit must be reduced to about 20. When more complex flow problems (for example, a supersonic jet impinging on a flat plate) were tried, the CFL requirements became unduly restrictive, and in some cases the program would not run without further code modifications. The primary modification was the addition of artificial dissipation. With such modifications the program solved all the problems attempted but in some cases required user interaction to "fine tune" CFL and dissipation parameters. Details of these modifications will be presented in the next section.

8.0 MODIFICATIONS

The purpose of this section is to describe modifications made/tried to improve the algorithm, both successful and not so successful.

The most successful modifications were in the artificial dissipation/smoother category. The first one considered was a finite volume analog of the term proposed by MacCormack (Ref. 3). The effect of this term is to increase the stability with essentially no change in the converged solution. The term uses the temporal changes and diminishes as a steady state is reached. It worked well, especially when starting up with poor initial guesses. The most successful artificial dissipation modification was a fourth-order term proposed by MacCormack (Ref. 7). The coefficient included both pressure and temperature derivatives, following Drummond (Ref. 8). This term increased the CFL restriction by almost an order-of-magnitude in some cases and allowed convergent solution of some problems that would not converge without the dissipation. Some caution is needed, as too much dissipation can significantly alter the solution obtained. Other smoothers were tried but resulted in little or no gain.

One of the failures was in the boundary-condition modifications for the implicit operator. Attempts to make these conditions more physically meaningful, for example,

$$\begin{aligned} \delta\varrho_b &= \delta\varrho \\ \delta E_b &= \delta E \\ \delta\varrho u_b &= \delta\varrho u \end{aligned} \tag{46}$$

did not work. Convergence problems increased with the new boundary conditions.

The biggest disappointment was the failure of an unfactored implicit operator to improve results. The approximate factorization, in addition to adding to the truncation error, is known to create stability problems, especially for 3-D problems (Ref. 9). Other implicit algorithms, because of their coupling, would require solution of large systems of equations with large bandwidth in an unfactored form. Because of its differencing approach, the implicit MacCormack algorithm in its unfactored form can be solved by simple backward substitution. This reduces the number of operations from $O(n^3)$ to $O(n^2)$. In fact, although the logic for programming the algorithm is significantly more difficult, the unfactored form can be solved just as efficiently as the factored form. Significant improvements were expected because the unfactored form kept a stronger coupling between the points. Unfortunately, the results for the unfactored form were almost identical with the factored form.

9.0 CONCLUDING REMARKS

The implicit MacCormack algorithm of Ref. 3 is a simple and efficient method. It can be easily added to any existing explicit MacCormack code. Its efficiency is attributable to: (1) the use of a block bidiagonal system instead of the more convectional tridiagonal system, (2) the ability to solve these systems in a scalar fashion using the characteristic transformation, and (3) the ability to bypass implicit calculations where local conditions permit.

Although unconditionally stable according to linearized 1-D theory, actual test cases have shown CFL limits and convergence difficulties when applied to complex flow problems. For simple flows, the algorithm converges quickly and gives excellent results.

Near the end of the code development/validation of KM1, the ARC code (Ref. 9) was acquired from NASA Ames. The ARC Code, when modified to solve the complex internal flow problems of interest, proved to be a robust and flexible code (PARC, Ref. 10). It has become the code of choice for the following reasons: it was available to the users earlier than the KM1 code, it is more robust in certain applications (better and faster convergence), and it is more user friendly. In essence, it provides the capability we expected from KM1. The PARC code has become the internal Navier-Stokes solver at AEDC, and there are no plans to convert the KM1 code to a production code.

A significant by-product of this effort has been the development of a characteristic-based boundary-condition algorithm. It has recently been added to other flow solvers, for example, the AEDC internal flow version of the NASA Ames Research Center ARC Code (Ref. 10), with significant improvements in results.

REFERENCES

1. Jacocks, J. L. and Kneile, K. R. "Computation of Three-Dimensional Time-Dependent Flow Using the Euler Equations." AEDC-TR-80-49 (AD-A102463), July 1981.

2. Koenig, K. and Barton, J. M. "Numerical Solution of the Three-Dimensional Unsteady Euler Equations." AEDC-TR-83-22 (AD-A132034), August 1983.

3. MacCormack, R. W. "A Numerical Method for Solving the Equations of Compressible Viscous Flow." AIAA Paper No. 81-0110, AIAA 19th Aerospace Sciences Meetings, St. Louis, Missouri, January 12 - 15, 1981.

4. Thomas, P. D. "Numerical Method for Predicting Flow Characteristics and Performance of Nonaxisymmetric Nozzles—Theory." NASA CR-3147, September 1979.

5. Thomas, P. D. "Numerical Method for Predicting Flow Characteristics and Performance of Nonaxisymmetric Nozzles. Part 2—Applications." NASA CR-3264, October 1980.

6. Mason, Mary L., Putnam, Lawrence E., and Re, Richard J. "The Effect of Throat Contouring on Two-Dimensional Converging-Diverging Nozzles at Static Conditions." NASA TP-1704, August 1980.

7. MacCormack, R. W. and Baldwin, B. S. "A Numerical Method for Solving the Navier-Stokes Equations with Application to Shock-Boundary Layer Interactions." AIAA Paper No. 75-1, AIAA 13th Aerospace Sciences Meeting, Pasadena, California, January 20 - 22, 1975.

8. Drummond, J. P. "Numerical Investigation of the Perpendicular Injector Flow Field in a Hydrogen Fueled Scramjet." AIAA Paper No. 79-1482, AIAA 12th Fluid and Plasma Conference, Williamsburg, Virginia, July 23 - 25, 1979.

9. Pulliam, T. H. "Euler and Thin Layer Navier-Stokes Codes: ARC2D, ARC3D." Notes for Computational Fluid Dynamics User's Workshop, University of Tennessee Space Institute (UTSI), Tullahoma, Tennessee, March 12 - 16, 1984, pp. 15.1 - 15.85.

10. Cooper, G. K. "Toward a General Purpose Navier-Stokes Code." *Abstracts, First World Congress on Computational Mechanics*, Vol. I, University of Texas, Austin, Texas, September 22 - 26, 1986.

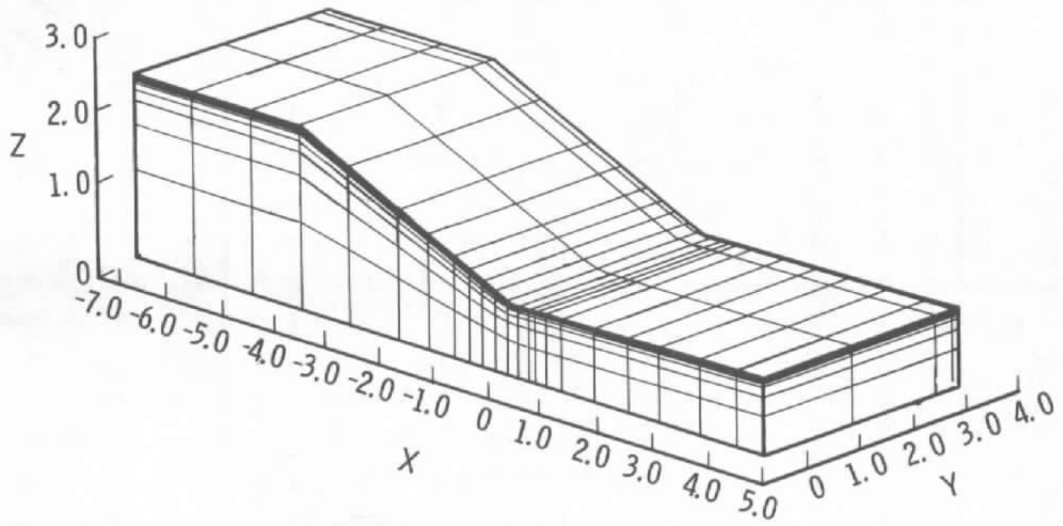

Figure 1. Grid for rectangular nozzle.

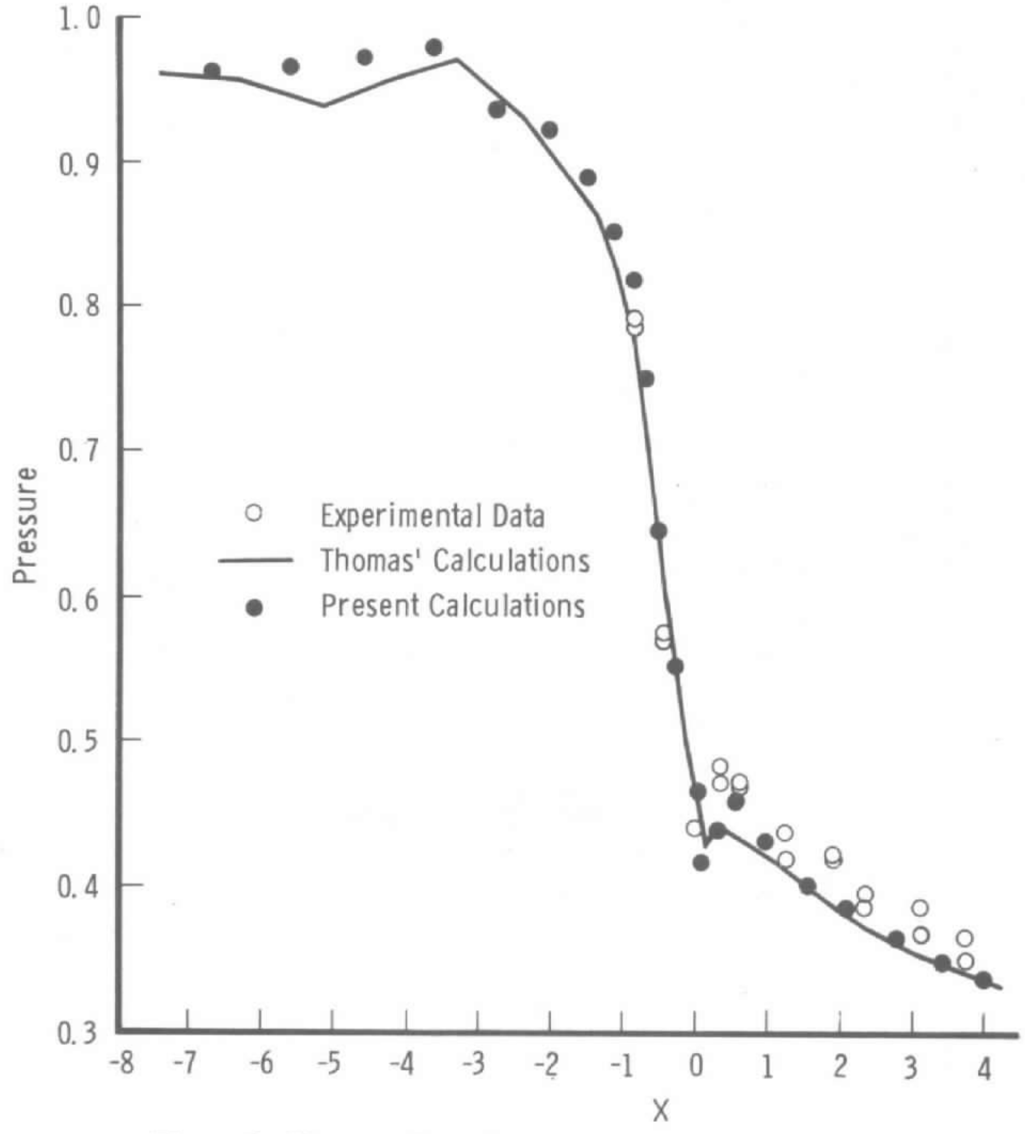

Figure 2. Upper wall static pressure at symmetry plane.

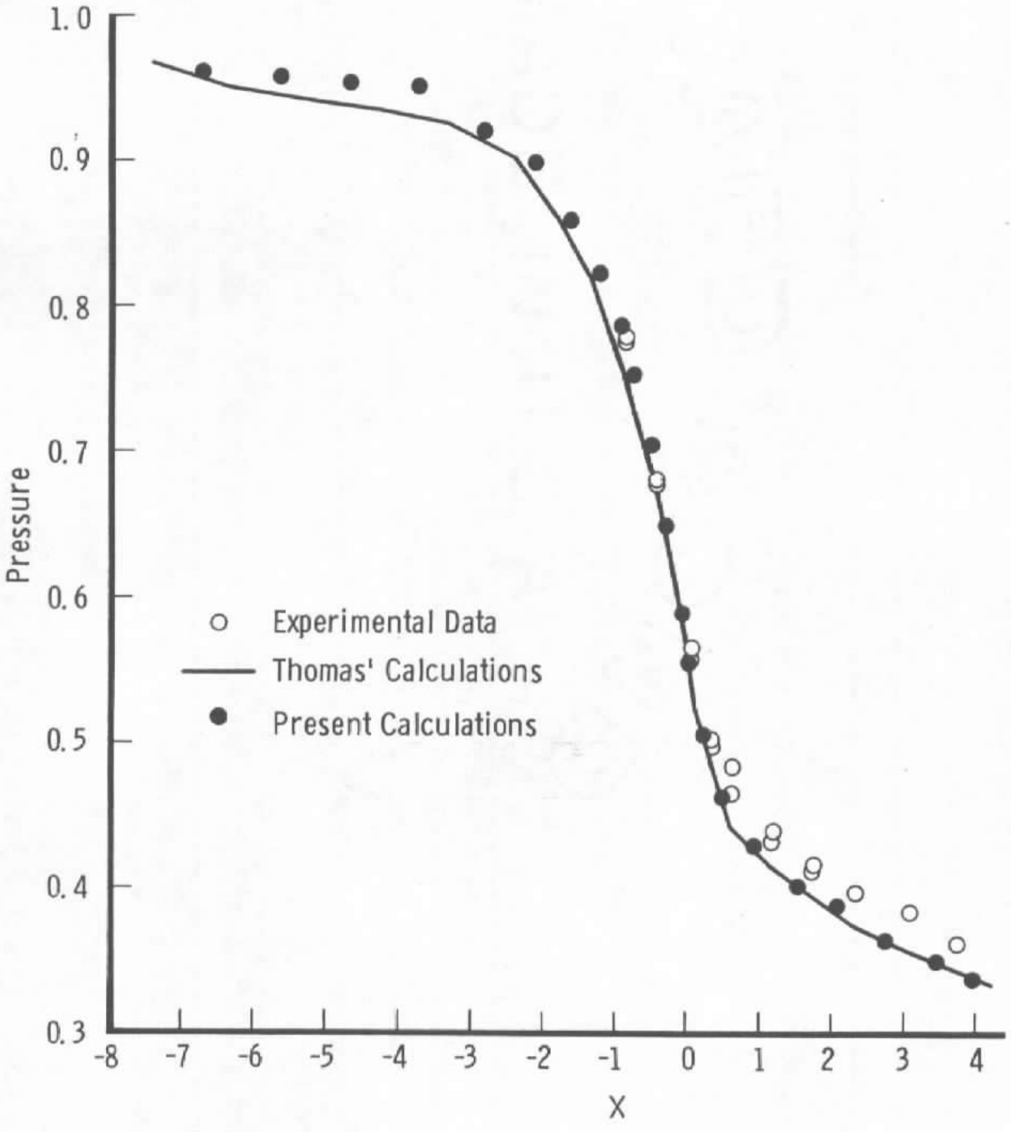

Figure 3. Side wall static pressure at symmetry plane.

NOMENCLATURE

A,B	Projected areas of two opposing cell faces (corresponds with subscript/superscript notation)
a	Speed of sound
c_p	Specific heat at constant pressure
D·	Implicit operator
E	Total energy per unit volume
F	Flux vector
F^*	Inviscid part of F
F_U	Flux Jacobian matrix, $\partial F/\partial U$
G	Viscous part of F
I	Identity (matrix, tensor, or dyadic)
k	Coefficient of thermal conductivity
L	Reference length
n	Unit normal
P	1) Pressure 2) Matrix factor of characteristic transformation
Pr	Prandtl number
Q	Characteristic transformation matrix
R	Matrix factor of characteristic transformation

Re	Reynolds number
S	Matrix factor of characteristic transformation
T	Temperature
t	Time
U	Conservative variables
u	Velocity vector
V	Volume of grid cell
α	$= u \cdot u/2$
β	$= \gamma - 1$
γ	Ratio of specific heats
Δ	Finite difference operator
Λ	Eigenvalue matrix
λ	Second coefficient of viscosity
μ	First coefficient of viscosity
ϱ	Density
Σ	Summation operator
τ	Stress tensor
ϕ	Null (matrix, tensor, or dyadic)
∇	Gradient operator
$\nabla\cdot$	Divergence operator

∇u	Symmetric part of ∇u
$\delta\tilde{U}$, $\delta\tilde{\tilde{U}}$	Intermediate values for factored implicit algorithm
ΔU	Right-hand side of implicit system after adding incoming fluxes
ΔU^*	Temporal changes at a boundary cell using interior points only
∂	Partial derivative notation
$\oint ds$	Surface integral over cell boundary

SUBSCRIPTS

A	Cell face where properties are evaluated using cell center values
B	Cell face where properties are evaluated using neighboring cell values
b	Phantom boundary points
i	First grid index
i^*	i or i+1 corresponding to backward/forward evaluation
j	Second grid index
k	Third grid index
r	Reference condition for Sutherlands law
w	Wall boundary condition
o	Total conditions
∞	Reference values for nondimensionalizing

SUPERSCRIPTS

A, B	Same meaning as subscript
1, 2, 3	Indices for 1-D implicit operator
-1	Inverse operator

SPECIAL NOTATIONS

D^+	Matrix of positive elements [see (Eq. 34)]
$\overline{n \cdot F_u}$	See Eq. (33)
$\lvert D \rvert$, $\lvert \Lambda \rvert$	Matrices containing absolute values of their elements